Michael G. Myers, Charles R. Christian,
Derrick K. Bennet, and Mario C. Murga

"DOCUMENT CONVERSION AND NETWORK DATABASE SYSTEM"

APPENDIX A

Appendix A is a hard copy printout of the assembly listing consisting of 37 pages, including this page.

```
'***
'***    Copyright Mario C. Murga, Health Informatics International, Inc. 1998
'***
'

Option Explicit
Public workingDirectory As String
Public targetFolder As String
Public conversionsFolder As String
Public footerExists As Boolean
Public bottomRule As Boolean
Public moduleName As String
Public moduleGif As String
Public moduleGif2 As String
Public moduleBack As String          'name of background gif
Public secondIndex As Boolean

Public currentFile As String         'so we only have to check once
Public currFileWindow As String
Public linkMapWindow As String
Public docMapWindow As String

Sub MainRoutine()

Application.DisplayScrollBars = False
'Application.DisplayStatusBar = False
Application.ScreenUpdating = False
'***
'***    This is the main routine that runs the whole conversion shebang
'***
'***    ignore (for now) the following files:
'***                                                         *.art
'***                                                    signon.*
'***                                                      menu.*
'***
    Dim currFile As String
    Dim namePart As String
    Dim extPart As String
    Dim saveName As String
    Dim messageText As String

    currFile = ""
    saveName = ""
    namePart = ""
    extPart = ""

    messageText = "Enter module code:" & vbCr & _
                    "AHA - Adult Health Advisor" & vbCr & _
                    "PA - Pediatric Health Advisor" & vbCr & _
                    "BHA - Behavioural Health Advisor" & vbCr & _
                    "WHA - Women's Health Advisor" & vbCr & _
                    "SHA - Senior Health Advisor" & vbCr & _
                    "CA - Cardiac Health Advisor" & vbCr & _
                    "OA - Ophthalmology Health Advisor" & vbCr & _
                    "SMA - Sports Medicine Advisor" & vbCr & _
                    "MA - Medications Advisor" & vbCr

    moduleName = InputBox(messageText, "Module Name Entry", "AHA")
```

```
tweakModuleName            'set some globals based on entered module name
workingDirectory = getArticlePath()'get working directory which contains raw articles
targetFolder = makeTargetFolder(moduleName) 'create folder for converted arts. and indexes
conversionsFolder = setConversionsFolder("Conversions")

Documents.Open _
    fileName:=docMap, _
    ConfirmConversions:=False, _
    addToRecentFiles:=False, _
    Format:=wdOpenFormatText, _
    ReadOnly:=True                    'open docmap file
Documents.Open _
    fileName:=linkMap, _
    ConfirmConversions:=False, _
    addToRecentFiles:=False, _
    Format:=wdOpenFormatText, _
    ReadOnly:=True                    'open linkmap file
currFile = Dir("") '(workingDirectory)      'retrieve first file
currFile = StrConv(currFile, vbLowerCase)

linkMapWindow = linkMap
docMapWindow = docMap

'   Debug.Print currFile
'   Debug.Print workingDirectory

Do While currFile <> ""          'null string is returned when no more files in folder
    bottomRule = False 'start out assuming no footer, will be set in footer/copyr. routine
    namePart = getNamePart(currFile) 'seperate the current filename into namepart and
extension part
    extPart = getExtPart(currFile)
    Select Case namePart               'first check filename cases
'    Case "credits"                    'skip conversion for credits file
    Case "signon"                      'skip conversion for signon file
    Case "menu"                        'skip conversion for menu file
    Case "linkmap"                     'skip conversion for linkmap.lmf
    Case "docmap"                      'skip conversion for docMap.dmf
    Case Else                          'no filename exceptions, check extension cases
        Select Case extPart
            Case "art"                 'skip coversion for art holder files
            Case "idx"                 'convert the current file as an index
                Documents.Open _
                    fileName:=currFile, _
                    ConfirmConversions:=False, _
                    addToRecentFiles:=False, _
                    Format:=wdOpenFormatText      'open current file for processing
                currentFile = ActiveDocument.Name  'get current filename sans path
                currFileWindow = currentFile      'so we can activate document windows
                prepIndex                         'convert as index file
            Case Else                             'all exceptions checked for
                Documents.Open _
                    fileName:=currFile, _
                    ConfirmConversions:=False, _
                    addToRecentFiles:=False, _
                    Format:=wdOpenFormatText      'open current file for processing
                currentFile = ActiveDocument.Name 'get current filename sans path
                currFileWindow = currentFile      'so we can activate document windows
```

```
                    convertArticle                      'ok to do the standard conversion
                    saveName = targetFolder & ":" & currFile & htmlHTML
'                    Debug.Print "targetFolder: " & targetFolder
'                    Debug.Print "currFile: " & currFile
'                    Debug.Print "saveName: " & saveName
                    With Documents(currFile)
                        .SaveAs fileName:=saveName, FileFormat:=wdFormatText
                        .Close
                    End With
            End Select
        End Select

        currFile = Dir                                  'get next file
        currFile = StrConv(currFile, vbLowerCase)
    Loop
    Documents(linkMap).Close
    Documents(docMap).Close

Application.DisplayScrollBars = True
'Application.DisplayStatusBar = True
Application.ScreenUpdating = True
End Sub

Private Function getArticlePath() As String
'***
'***    The purpose of this function is to return the path
'***    for the articles to be converted
'***
'
    With Dialogs(wdDialogFileOpen)
        .Show
    End With

    '***
    '***    it doesn't matter if a file is opened or not,
    '***    CurDir returns the last navigated path
    '***    return value to calling
    '***
    getArticlePath = CurDir
End Function

Public Function getNamePart(ByVal fileName As String) As String
'***
'***    Return the name part of a filename
'***
    Dim dotLoc As Long
    dotLoc = InStr(fileName, ".")
    getNamePart = Left(fileName, dotLoc - 1)
End Function

Public Function getExtPart(ByVal fileName As String) As String
'***
'***    Return the extension part of a filename
'***
    Dim dotLoc As Long

    dotLoc = InStr(fileName, ".")
    getExtPart = Right(fileName, Len(fileName) - dotLoc)
End Function

Public Function makeTargetFolder(ByVal targetFolder As String) As String
'***
'***    MkDir ==> create a folder based on the current path
'***
'***    The "::" part makes it go up one level before creating the directory
'***
Dim saveDir As String

    MkDir "::" & targetFolder      'create target folder on the same level as working folder
    saveDir = CurDir               'save current folder to return to
    ChDir "::" & targetFolder      'go to newly created folder
    makeTargetFolder = CurDir  'make the return value of this function be the created folder
    ChDir saveDir                  'go back to working folder
End Function

Public Function setConversionsFolder(ByVal cFolder As String) As String
'***
'***    MkDir ==> create a folder based on the current path
'***
'***    The "::" part makes it go up one level before creating the directory
'***
Dim saveDir As String

    saveDir = CurDir         'save current folder to return to
    ChDir "::" & cFolder         'go to newly created folder
    setConversionsFolder = CurDir 'make the return value of this funct. be the created folder
    ChDir saveDir                         'go back to working folder
End Function
'=====================================================================
Public Sub convertArticle()
'
'***
'***    This file was created using FormatSpecialA thru D.
'***    These modules must be run in this order.
'***    Basically, do the article conversion
'***
'
    multiReplace conversionsFolder & ":" & commentTags  'change raw tags to commented HTML
    specialText "[", "]", htmlBoldStart, htmlBoldEnd, True, True   'replace [...] with
<b>...</b>
    CreateTitle                                 'make window and article title
    HideHeaderInfo                              'hide the med codes
    FooterCopyright                             'prettify copyright info and nix CRS
    PreformattedListsandTables                  'add preformatting tags to tables and menus
'
'***
'***    Note that prior to running Step 3.b, two files named
'***    "linkmap.lmf" and "docmap.dmf" needed to be available.
'***
'
    DocumentAnchor                        'make links to other docs and pics
    SectionLinks                          'make links at top to inner sections
    MakeParagraphs                        'format paragraphs
    MakeLists                             'format bulleted and numbered lists
    InsertTopText                           'insert top text template
    InsertBottomText                        'insert bottom text template and tailor to module
End Sub
'=========================================================================
Option Explicit

Public Sub CreateTitle()
'***
'***    This routine finds the delimited article title and uses it
'***    to create the window title and the displayed article title.
'***    Assumes tagged ascii format.
'***    This routine also adds the top and bottom tokens to the article.
'***
    Dim docTitle As String
    Dim windowTitle As String
    Dim missEnd As Boolean
    Dim gotWindowTitle As Boolean
    '***
    docTitle = missingTitleDebug            'debugging title in case none found
    windowTitle = missingTitleDebug         'debugging title in case none found
    missEnd = False
    '***
    Selection.HomeKey unit:=wdStory, Extend:=wdMove  'move to start of doc
    SimpleFind begTitleText                          'find first title
    Do While Selection.Find.Found
        Selection.MoveDown unit:=wdParagraph 'move to line following begTitleText - assume it
is doc title
        Selection.ExtendMode = True                  'selection mode on
        SimpleFind endTitleText                      'selection includes title and end tag
        If Selection.Find.Found Then
            Selection.End = Selection.End - Len(endTitleText) 'deselect endTitletext leaving
only title selected
            Selection.End = Selection.End - 1                 'get rid of spurious cr
        Else                                 'jumpin jehosifat! no endTitleText tag found.
            missEnd = True                   'Assume title is one paragraph.
            Selection.MoveDown unit:=wdParagraph, Extend:=wdExtend 'move to end of hoped for
title
        End If
        docTitle = Selection.Text                         'save the title for cleaning
        If Not gotWindowTitle Then                        'only do if no windo title yet
            windowTitle = stripDelimiters(docTitle, paraDelimiter, " ") 'remove par. delimiter
            gotWindowTitle = True          'show that window title is already accounted for
        End If
        docTitle = stripDelimiters(docTitle, paraDelimiter, htmlBreak) 'replace paragraph
delimiter with html break
        Selection.ExtendMode = False
        Selection.Text = htmlStartTitle & docTitle & htmlEndTitle 'replace selected text with
cleaned title with html tags
        Selection.Collapse wdCollapseEnd
        SimpleFind begTitleText            'find next title (subtitle in spanish files)
    Loop

    SimpleFind endTitleText                               ' find end tag again
    If Selection.Find.Found Then
        Selection.Collapse wdCollapseEnd
        Selection.Text = vbCr & afterTitleTag & vbCr & TopLinkTag 'leave a tags after last
title
```

```
        Selection.Collapse wdCollapseEnd
    End If
    Selection.HomeKey unit:=wdStory         'go back and insert top stuff for HTML & title
    Selection.Text = docTopToken & vbCr & htmlStartHead & _
                    windowTitle & htmlEndHead & vbCr 'insert token for start of doc and
actual window title
    Selection.Collapse wdCollapseEnd
    If missEnd Then
        Selection.InsertAfter titleEndDebug & vbCr            'insert debug string
        Selection.Collapse wdCollapseEnd
        Selection.Text = vbCr & afterTitleTag & vbCr & TopLinkTag 'leave a tags after last
title
    End If
    Selection.EndKey unit:=wdStory 'go to bottom and insert bottom stuff for HTML & footer
    Selection.Text = vbCr & docBottomToken & vbCr             'insert token for end of doc
    Selection.Collapse wdCollapseEnd
End Sub
'=====================================================================
Option Explicit

Public Sub DocumentAnchor()
'***
'*** Modify document such that index targets are true html anchors and all hyperlinks are
true hyperlinks
'*** This macro must be run on files with original names, i.e no ".html" extension
'***
    Dim currentSection As String                'linked title of section
    Dim currentSectionText As String            'body text between vTag and qTag
    Dim cst As String                           'abbreviation of currentSelectiontext
    Dim braceLoc As Long                        'location of begLink tag in cst
    Dim targetLetter As String                  'letter between "" & "", link target!
    Dim currSectTarget As String                'currentSection plus targetLetter
    Dim htmlLink As String                      'resulting html link for targetLetter
    '***
    '***    Initialize variables
    '***
    '***    assumes every section title needs to also be an anchor
    '***

    Selection.HomeKey unit:=wdStory             'goto top of doc to know where we are
    SimpleFind cTag                             'find first (usually only) article start
    Do While Selection.Find.Found
        Selection.MoveDown unit:=wdParagraph  'move to line following cTag - assume it is
section title
        Selection.Text = startAnchorToken     'insert token for start anchor definition
        Selection.Collapse wdCollapseEnd
        Selection.MoveDown unit:=wdParagraph, Extend:=wdExtend         'select rest of
        Selection.End = Selection.End - 2     'deselect  leaving only  title selected
        Selection.Range.Case = wdLowerCase 'change selection case to lower so links will work
        currentSection = Selection.Text       'save the  title
        Selection.Collapse wdCollapseEnd      'move to end of  and deselect
        Selection.InsertAfter endAnchorToken & vbCr 'insert token for end anchor definition
        Selection.Collapse wdCollapseEnd
        '
        SimpleFind vTag                               'goto end of section header
        If Not Selection.Find.Found Then Selection.InsertAfter vTagDebug & vbCr 'hey, no vTag!
        Selection.MoveDown unit:=wdParagraph          'goto start of actual article body
    Selection.ExtendMode = True
    SimpleFind qTag                          'selection includes entire current section body
    If Not Selection.Find.Found Then                        'now what, no qTag?
        Selection.ExtendMode = False
        Selection.InsertAfter qTagDebug & vbCr              'insert qTag debug message
        Selection.Collapse wdCollapseEnd
        Selection.ExtendMode = True
        Selection.EndKey unit:=wdStory                      'fake it by moving to end
    End If
    Selection.ExtendMode = False 'end extend selection so next find won't change selection
    currentSectionText = Selection.Text  'put section text into a string to be manipulated


'***
'***    This next routine looks through the current section text, which is selected at this
'***    point, and embeds HTML links to any any sections referenced between braces, i.e. I, or
'***    A, etc. It will do this by copying the selection into a string variable and searching
'***    for occurances of begLink and endLink, currently:
'***    "" - any single character - "". The "" and "" characters replaced the "<" and ">" '***
'***    characters in the mass replacements routine to not conflict with embedded html tags.
'***    The single character between the braces will be used along with currentSection$ to
'***    pull the target filename and section name out of the Link Map and Document Map arrays.
'***    This action will be repeated until no more ? strings are found.
'***    Processing will then proceed with the next section, etc.
'***    Note that references that return a file named *.art will be handled specially since
'***    this is where the GIF thumbnail and link to the JPEG picture is inserted
'***


    cst = currentSectionText                    'needed a shorter name to work with
    braceLoc = InStr(cst, begLink)              'find location of "" which starts "?"
    Do While braceLoc > 0                       'if a brace was found
        If Mid(cst, braceLoc + 2, 1) = endLink Then   'if second following char is ""
            targetLetter = LCase(Mid(cst, braceLoc + 1, 1)) 'valid single index letter was
found
            currSectTarget = currentSection + "~" & targetLetter   'build target for link
map array
            htmlLink = extractLink(currSectTarget)   'extract target section from link map
array & file map array
            cst = Left(cst, braceLoc - 1) & _
                    htmlLink & _
                    Right(cst, (Len(cst) - (braceLoc + 2)))   'build new section text
including HTML link in place of "?"
                    'remember: htmlLink might be debug text
        Else    'must have originally been an isolated "<" symbol
            cst = Left(cst, braceLoc - 1) & _
                    htmlLessThan & _
                    Right(cst, (Len(cst) - braceLoc)) 'put the less than symbol back in
        End If

'***
'***    start search after last "" found in case there are braces
'***    not of the "?" form, so it won't infinite loop
'***
        braceLoc = InStr(braceLoc + 1, cst, begLink)      'find location of next ""
    Loop
    currentSectionText = cst                    'go back to longer name
    Selection.Text = currentSectionText         'replaces currently selected section
                                                'text with newly built linked version
        Selection.Collapse wdCollapseEnd
        SimpleFind cTag                              'find next section
    Loop
End Sub

'*************************************************************************
'***  Extract target section from link map array & file map array
'*************************************************************************

Private Function extractLink(ByVal target As String) As String

    Dim currSection As String
    Dim linkSection As String
    Dim linkDoc As String
    Dim art As String
    Dim dot As Long
    Dim jpegName As String
    Dim gifName As String

    currSection = currentFile & "~" & target       'prepend current file to current section
                                                   'name for use in the link from array
    linkSection = sectionToSection(currSection)    'call to search link map file
    If linkSection = missingLinkMapDebug Then      'check for missing linkmap target
        extractLink = linkSection                  'return debug message
        Documents(currFileWindow).Activate         'make sure we re-activate working window
        Exit Function
    End If
    linkDoc = sectionToDoc(linkSection)            'call to read doc map file
    If linkDoc = missingDocMapDebug Then           'check for missing docmap target
        extractLink = linkDoc                      'return debug message
        Documents(currFileWindow).Activate         'make sure we re-activate working window
        Exit Function
    End If
    Documents(currFileWindow).Activate             'make sure we re-activate working window
    art = Right(linkDoc, 3)                        'Check to see if file to link to
                                               'is of the form "*.art" for special handling
    If art = "art" Then                        'this is an art ink
        dot = InStr(linkDoc, ".")
        If dot > 1 Then linkDoc = Left(linkDoc, dot - 1)  'take only the file name sans ext.
        jpegName = linkDoc & ".jpg"                       'build jpeg filename
        gifName = linkDoc & ".gif"                        'buld gif filename
        extractLink = htmlBreak & htmlAStart & qq & artDir & jpegName & qq & _
                      htmlET & htmlImgStart & qq & artDir & gifName & qq & _
                      htmlImgEnd & htmlPressHere & htmlAEnd & htmlBreak
                                                          'this is a regular link
    Else
        extractLink = htmlBreak & htmlAStart & qq & linkDoc & htmlHTML & ttt & _
                      linkSection & qq & htmlET & htmlGoThere & htmlAEnd
    End If
End Function

'*************************************************************************
'***    reads link map  file to find target section
'*************************************************************************

Private Function sectionToSection(ByVal cs As String) As String
'***
'***    cs  = filename + "~" + target letter
```

```
'***    ts  = target section
'***    sts = line containing target section
'***
    Dim ts As String
    Dim sts As String
    Dim firstTilde As Long
    Dim secondTilde As Long

    Documents(linkMapWindow).Activate                 'activates linkmap doc window
    ts = ""
    sts = ""

    Selection.HomeKey unit:=wdStory                   'go to top of doc

    SimpleFind cs                                     'goes right to the proper line
    If Selection.Find.Found Then
        Selection.ExtendMode = True
        Selection.MoveDown unit:=wdParagraph          'extend selection to end of paragraph
        Selection.End = Selection.End - 1             'don't include  mark
        Selection.ExtendMode = False
        sts = Selection.Text                  'put selection into a string to be manipulated
        firstTilde = InStr(sts, "~")                  'find location of 1st "~"
        secondTilde = InStr(firstTilde + 1, sts, "~") 'find location of 2nd "~"
        ts = Right(sts, (Len(sts) - (secondTilde + 2)))          'extract target section
    Else
        ts = missingLinkMapDebug                      'target not found
    End If

    sectionToSection = ts                             'return target section
    Documents(currFileWindow).Activate                'activates current article doc window
End Function

'*********************************************************************
'reads doc map  file to find target section
'*********************************************************************

Private Function sectionToDoc(ByVal ls As String) As String
'***
'***    ls = link section returned by sectionTosection function
'***
    Dim td As String                                  'extracted target section
    Dim std As String                                 'line containing target section
    Dim firstTilde As Long

    Documents(docMapWindow).Activate                  'activate docmap doc
    td = ""
    std = ""

    Selection.HomeKey unit:=wdStory                   'start at top of document

    SimpleFind ls                             'goes right to the proper line
    If Selection.Find.Found Then
        Selection.ExtendMode = True
        Selection.MoveDown unit:=wdParagraph          'extend selection to end of paragraph
        Selection.End = Selection.End - 1             'don't include  mark
        Selection.ExtendMode = False
        std = Selection.Text                  'put selection into a string to be manipulated
        firstTilde = InStr(std, "~")                    'find location of 1st "~"
        td = Right(std, (Len(std) - firstTilde))        'extract target section
    Else
        td = missingDocMapDebug                         'target not found
    End If
    sectionToDoc = td                                   'return target
    Documents(currFileWindow).Activate                  'activates current article doc window
End Function

'=================================================================
Option Explicit
Public Sub FooterCopyright()
    footerSection crTag                                         'do "/copyright" case
    footerSection fTag                                          'do ".F" case
    deleteCRSCopyright                                          'strip CRS Copyright

    If Not bottomRule Then       'add horizontal rule if not added in footer routiners above
        Selection.HomeKey unit:=wdStory                        'always know where you are (top)
        SimpleFind qTag
        If Selection.Find.Found Then
            Selection.Collapse wdCollapseStart                  'insert in front of qTag
            Selection.Text = htmlHorizontalRule & vbCr          'add horizontal rule rule
            bottomRule = True                                   'we added a bottom rule
        Else
            SimpleFind docBottomToken                           'find doc bottom a different way
            If Selection.Find.Found Then
                Selection.Collapse wdCollapseStart              'insert in front of docBottomToken
                Selection.Text = htmlHorizontalRule & vbCr          'add horizontal rule rule
                bottomRule = True                               'we added a bottom rule
            End If          'hey if we can't find the bottom at this point we're in trouble
        End If
        Selection.Collapse wdCollapseEnd
    End If
End Sub

Public Sub footerSection(footerType As String)
    '***
    '*** beautify article copyright footer section
    '***
    Selection.HomeKey unit:=wdStory                             'always know where you are (top)
    SimpleFind footerType                                       'find footer tag
    Do While Selection.Find.Found                               'works for either
        bottomRule = True                   'set global flag to prevent double <hr> at bottom
        Selection.Text = footerType & htmlStartArticleFooter   'start of footer section HTML
        Selection.Collapse wdCollapseEnd
        SimpleFind qTag                                     'go to end of article copyright footer
        If Selection.Find.Found Then
            Selection.ExtendMode = False                        'end extend selection
            Selection.Collapse direction:=wdCollapseStart 'insertion point at start of .Q tag
            Selection.InsertAfter htmlEndArticleFooter & vbCr       'end of footer HTML
            Selection.Collapse wdCollapseEnd
        Else
            Selection.EndKey unit:=wdStory                       'move to end to add end Q tag
            Selection.InsertAfter htmlEndArticleFooter & qTagDebug & vbCr 'end of footer HTML
(hopefully!)
            Selection.Collapse wdCollapseEnd
        End If
        SimpleFind footerType                               'satisfy while condition
    Loop
End Sub

Public Sub deleteCRSCopyright()
    '***
    '***    Strip CRS copyright, if it exists, since it is added later
    '***

    Selection.HomeKey unit:=wdStory                          'always know where you are (top)
    PatternFind wildCopyRight                   'uses ???? pattern matching to allow all years

    If Selection.Find.Found Then
        Selection.MoveUp unit:=wdParagraph      'go to beginning of selection paragraph
        Selection.Range.Delete                               'delete ^p before
        Selection.MoveDown unit:=wdParagraph, Extend:=wdExtend     'select para with copyright
        Selection.Text = deletedCopyrightDebug               'replace copyright with debug tag
        Selection.Collapse wdCollapseEnd
        Selection.ExtendMode = False                         'cancel extend mode
    End If
End Sub

'=====================================================================
Option Explicit
Public Sub HideHeaderInfo()
'***
'***    This rountine comments out the header information of
'***    the article which includes the med codes.
'***    Assumes there is a link anchor on the line following the cTag
'***

    Selection.HomeKey unit:=wdStory                  'always know where you are (top)
    SimpleFind cTag                          'assume it is there
'     Debug.Print "HideHeaderInfo Find cTag selection.find.found = " & Selection.Find.Found
    Do While Selection.Find.Found
        Selection.MoveDown unit:=wdParagraph, Count:=2     'move to line following anchor
        Selection.InsertAfter htmlStartComment     'start of HTML comment
        Selection.Collapse wdCollapseEnd
        SimpleFind vTag                          'find known end of header info

'     Debug.Print "HideHeaderInfo vTag  = " & vTag
'     Debug.Print "HideHeaderInfo Find vTag selection.find.found = " & Selection.Find.Found
        If Selection.Find.Found Then                  'make sure there is a vTag
            Selection.MoveLeft Count:=2     'deselect vTag leaving only header info selected
            Selection.InsertAfter htmlEndComment & vbCr  'end of HTML comment plus a newline
            Selection.Collapse wdCollapseEnd
        Else                                         'uh-oh, no vTag!
            Selection.InsertAfter htmlEndComment & vTagDebug & vbCr 'insert a debugging
comment, vTag missing
            Selection.Collapse wdCollapseEnd
        End If
        SimpleFind cTag                          'in case there is another section - satisfy while
    Loop
End Sub
'======================================================================
Option Explicit
'***
```

```
'***`   These constants are used in searches
'***
Public Const crTag          As String = "<!-- /copyright -->"
Public Const fTag           As String = "<!-- .F -->"
Public Const qTag           As String = "<!-- .Q -->"
Public Const cTag           As String = "<!-- .C -->"
Public Const vTag           As String = "<!-- .V -->"
Public Const hpTag          As String = "/hp"            'primary heading tag
Public Const pTag           As String = "/p"             'paragraph tag
Public Const bliTag         As String = "/bli "          'bulleted list item tag
Public Const numTag         As String = "/numitem "      'numbered list item tag
Public Const letTag         As String = "/letteritem "   'letter list item tag
Public Const romTag         As String = "/letteritem "   'roman numeral list item tag
'
Public Const pCommentTag    As String = "<!-- /p 0/0 -->"
Public Const hpCommentTag   As String = "<!-- /hp -->"
Public Const hpLastTag      As String = "<!-- hp last -->"
Public Const TopLinkTag     As String = "<!-- top link -->"
'
Public Const begTitleText As String = "<!-- /btt -->"
Public Const endTitleText As String = "<!-- /ett -->"
'
Public Const beg2ColList  As String = "<!-- /btwocollist -->"
Public Const end2ColList  As String = "<!-- /etwocollist -->"
'
Public Const begBookList  As String = "<!-- /bbooklist -->"
Public Const endBookList  As String = "<!-- /ebooklist -->"
'
Public Const begMenu  As String = "<!-- /bmenu -->"
Public Const endMenu  As String = "<!-- /emenu -->"
'
Public Const begTable  As String = "<!-- /btable -->"
Public Const endTable  As String = "<!-- /etable -->"
'
Public Const begLink  As String = ""
Public Const endLink  As String = ""
'
Public Const paraDelimiter     As String = ""
Public Const linkMap           As String = "linkmap.lmf"
Public Const docMap            As String = "docmap.dmf"
'
Public Const topText           As String = "top.txt"  'where the top text template is
Public Const bottomText        As String = "bottom.txt"  'where the bottom text template is
Public Const index1Return      As String = "index1Return.txt"  'where the index 1 return link
text template is
Public Const index2Return      As String = "index2Return.txt"  'where the index 2 return link
text template is
Public Const mainIndexReturn   As String = "mainIndexReturn.txt"  'where the main index return
link text template is
Public Const HIILegal          As String = "HIILegal.txt"   'where the HII legal text template
is
'
Public Const index1Token       As String = "<!-- index1Return -->"  'where the index 1 return
link token
Public Const index2Token       As String = "<!-- index2Return -->"  'where the index 2 return
link token
Public Const mainIndexToken    As String = "<!-- mainIndexReturn -->" 'where the main index
```

```
return link token
Public Const HIILegalToken      As String = "<!-- HIILegal -->" 'where the HII legal text token
Public Const moduleToken        As String = "$module$" 'where the HII legal text token
Public Const moduleToken2        As String = "$module2$" 'where the HII legal text token
Public Const moduleBackToken     As String = "$moduleback$"    'where the background image is
referenced"
'
Public Const commentTags         As String = "commentTags"  'assume Conversions folder at same
level as article folder
Public Const afterTitleTag       As String = "<!-- after title -->" 'used to show location after
last title
Public Const tokensToHTML        As String = "tokensToHTML"  'assume Conversions folder at same
level as article folder
Public Const addTopAndBottom     As String = "addTopAndBottom"  'assume Conversions folder at
same level as article folder
'***
'***    These constants are used as inserted text
'***
Public Const htmlParaStart              As String = "<p>"
Public Const htmlParaEnd                As String = "</p>"
'
Public Const htmlBQStart              As String = "<blockquote>"
Public Const htmlBQEnd                As String = "</blockquote>"
'
Public Const htmlPreStart              As String = "<pre>"
Public Const htmlPreEnd                As String = "</pre>"
'
Public Const htmlBoldStart              As String = "<b>"
Public Const htmlBoldEnd                As String = "</b>"
'
Public Const htmlStartArticleFooter As String = "<!-- start of footer --><BR><HR><BR><EM>"
Public Const htmlEndArticleFooter   As String = "</EM><P><HR><!-- end of footer -->"
'
Public Const htmlStartComment       As String = "<!-- "
Public Const htmlEndComment         As String = " -->"
'
Public Const htmlStartHead          As String = "<!-- start of header --><HEAD><TITLE>HII - "
Public Const htmlEndHead            As String = "</TITLE></HEAD><!-- end of header -->"
'
Public Const htmlStartTitle         As String = "<H1 align=center><B>"
Public Const htmlEndTitle           As String = "</B></H1>"
Public Const htmlTopLinkStart As String = "<br><img src=" & qq & "../images/bullet.gif" & qq &
_                                             " align=TOP width=13 height=13><a href=" & qq &
"#"   'start of links at top of article
'
Public Const htmlBreak              As String = "<br>"
Public Const htmlHorizontalRule     As String = "<hr>"
Public Const htmlLessThan           As String = "&lt;"
Public Const htmlGreaterThan        As String = "&gt;"
'
Public Const htmlAStart             As String = "<A href="
Public Const artDir                 As String = "art/"
Public Const htmlET                 As String = ">"
Public Const htmlImgStart           As String = "<IMG SRC="
Public Const htmlImgEnd             As String = " align=center vspace=30 hspace=30>"
Public Const htmlPressHere          As String = "<font size=5><B>Press here to view a full
```

```
size picture.</B></font>"
Public Const htmlAEnd                  As String = "</A>"
Public Const htmlGoGif                 As String = "go.gif"
Public Const htmlAlt                   As String = " alt="
Public Const htmlGo                    As String = "(Go There)"
Public Const htmlHTML                  As String = ".html"
Public Const htmlNoBorder              As String = " border=0>"
Public Const ttt                       As String = "#"
Public Const htmlGoThere               As String = "<img src=""go.gif"" alt=""Go There""
border=0>"
Public Const qq                        As String = """"   'double quote
'
Public Const docTopToken               As String = "<!-- top of article -->"
Public Const docBottomToken            As String = "<!-- bottom of article -->"
'
Public Const startAnchorToken          As String = "<p><A Name=" & qq
Public Const endAnchorToken            As String = qq & "></a>"
Public Const wildCopyRight             As String = "Copyright [1-2][0-9][0-9][0-9] Clinical
Reference Systems"        'note ???? wildcard for year

'***
'***    ASP Includes
'***
Public Const ASPdocmain      As String = "<!--#include virtual=" & qq & "/include/docmain.inc"
& qq & "-->"
Public Const ASPdochdr1      As String = "<!--#include virtual=" & qq & "/include/dochdr1.inc"
& qq & "-->"
Public Const ASPdochdr2      As String = "<!--#include virtual=" & qq & "/include/dochdr2.inc"
& qq & "-->"
Public Const ASPdochdr3      As String = "<!--#include virtual=" & qq & "/include/dochdr3.inc"
& qq & "-->"

'***
'***    Inserted debugging tags
'***
Public Const endTagDebug                  As String = "<!-- Debug: Fake endTag -->"
Public Const vTagDebug                    As String = "<!-- Debug: Missing vTag -->"
Public Const qTagDebug                    As String = "<!-- Debug: Missing qTag -->"
Public Const pTagDebug                    As String = "<!-- Debug: Missing pTag -->"
Public Const missingParaDelimiterDebug As String = "<!-- Debug: Missing Paragraph End
Delimiter -->"
Public Const missingTitleDebug            As String = "<!-- Debug: Missing Article Title Start
Tag -->"
Public Const titleEndDebug                As String = "<!-- Debug: Missing Article Title End Tag
-->"
Public Const deletedCopyrightDebug        As String = "<!-- Debug: Deleted Copyright -->"
Public Const missingLinkMapDebug          As String = "<!-- Debug: Missing target in linkmap.lmf
-->"
Public Const missingDocMapDebug           As String = "<!-- Debug: Missing target in docmap.dmf
-->"
'

Public Sub stripEndReturns()
    Dim allClear As Boolean

    allClear = False                              'initialize
    Selection.EndKey unit:=wdStory, Extend:=wdMove     'move to end of document
    Do
        Selection.MoveLeft Extend:=wdExtend    'select last character
        If (Selection.Text = vbCr) Or (Selection.Text = vbCrLf) Then
            Selection.Text = ""
            Selection.Collapse wdCollapseStart
        Else
            allClear = True
            Selection.ExtendMode = False       'cancel selection extend
        End If
    Loop Until allClear
    Selection.HomeKey unit:=wdStory, Extend:=wdMove    'move to start of document
End Sub

'***
'*** Special handling for text which begins with startText
'*** and ends with endText
'*** Enclose this text with  HTML codes htmlOpen and htmlClose
'*** deleteFirst is true if first character should be deleted (i.e a "[")
'*** deleteLast is true if last character should be deleted (i.e a "]")
'***

Public Sub specialText(ByVal startText As String, ByVal endText As String, _
                       ByVal htmlOpen As String, ByVal htmlClose As String, _
                       ByVal deleteFirst As Boolean, ByVal deleteLast As Boolean)

    Selection.HomeKey unit:=wdStory                   'move to start of doc
    SimpleFind startText                              'find start of special text - if any
    Do While Selection.Find.Found

        'move to beginning of selected text if startText not to be deleted
        If Not deleteFirst Then Selection.Collapse direction:=wdCollapseStart

        Selection.Text = htmlOpen          'HTML start text token, overwrites if deleteFirst
        Selection.Collapse wdCollapseEnd
        SimpleFind endText                           'find end delimiter
        If Selection.Find.Found Then
            If Not deleteLast Then
                Selection.Collapse direction:=wdCollapseEnd    'move to end of selected text
if endText not to be deleted
            End If
            Selection.Text = htmlClose               'HTML end text token
            Selection.Collapse wdCollapseEnd
        Else
            Selection.EndKey                 'move to end of line, at least do one line
            Selection.Text = htmlClose
            Selection.Collapse wdCollapseEnd
        End If

        SimpleFind startText                          'find next start of special text
    Loop
End Sub

Public Sub preformattedText(ByVal startTag As String, ByVal endTag As String)
'***
'***    This routine requires a start tag and end tag.
'***    The start tag is used to find and mark the beginning of
'***    preformatted text blocks.
```

```
'***   The end tag is used to find and mark the end of
'***   preformatted text blocks.
'***   The global constants htmlPreStart and htmlPreEnd are used.
'***   The are expected to be <pre> and </pre>, but don't have to be.
'***
    Dim docText As String                       'stores text to be cleaned of  delimiter
    Dim paraLoc As Long                         'place keeper for location of  delimiter
    Dim tagLen As Long                          'used to subtract endTag from selection
    Dim fakeEnd As Boolean                      'keep track of when an endTag is forced
    '***
    fakeEnd = False                             'initialize
    '***
    Selection.HomeKey unit:=wdStory             'always know where you are (top)
    SimpleFind startTag                         'find start of list if any
    Do While Selection.Find.Found
        Selection.Text = startTag & htmlPreStart  'start of preformatted list
        Selection.MoveDown unit:=wdParagraph      'move to line following tag
        Selection.ExtendMode = True
        SimpleFind endTag                         'selection includes table
        If Selection.Find.Found Then              'legit endTag was found
            tagLen = Len(endTag)
            fakeEnd = False
        Else                                      'force an end is none found
            SimpleFind fTag                       'look for article footer
            If Selection.Find.Found Then
                tagLen = Len(fTag)
                fakeEnd = True
            Else                                  'fTag was not found, keep looking
                SimpleFind crTag                  'look for article copyright
                If Selection.Find.Found Then
                    tagLen = Len(crTag)
                    fakeEnd = True
                Else                              'crTag was not found, use last resort
                    Selection.EndKey unit:=wdStory, Extend:=wdExtend  'extend to end of
article as last resort
                    tagLen = 0                     'dont really have a tag
                    fakeEnd = True
                End If
           End If
        End If

        Selection.ExtendMode = False               'end extend selection
        Selection.End = Selection.End - tagLen     'deselect tag leaving only text selected
        docText = Selection.Text                   'save the text for cleaning
        paraLoc = InStr(docText, paraDelimiter)    'find location of ""
            Do While paraLoc > 0                   'only bother if there is a ""
                docText = Left(docText, paraLoc - 1) & _
                          Right(docText, Len(docText) - paraLoc)          'strip ""
                paraLoc = InStr(docText, paraDelimiter)          'find location of next ""
            Loop

        Selection.Text = docText                   'replace selected text with cleaned text
        Selection.Collapse direction:=wdCollapseEnd 'collapse insertion point to start of end
tag
        Selection.Text = htmlPreEnd                 'end preformatted list
        Selection.Collapse wdCollapseEnd
        If fakeEnd Then
```

```
            Selection.Text = endTagDebug            'add special tag to allow troubleshooting
            Selection.Collapse wdCollapseEnd
        End If
        SimpleFind startTag                                'look for more preformatted text
    Loop
End Sub

Public Function stripDelimiters(ByVal docText As String, _
                                ByVal delimiterText As String, _
                                Optional ByVal replaceIt As String = "") As String
'***
'***    This function takes the passed docText string and replaces all occurances of
'***    delimiterText passed to it with the string replaceIt. If the optional string replaceIt
'***    is not passed, the delimiter is deleted rather than replaced.
'***
    Dim delLoc As Long                                  'delimiter location in string
    Dim delLen As Long                                  'lenth of delimiter text
    '
    delLen = Len(delimiterText)                         'get delimiter text length
    '
    delLoc = InStr(docText, delimiterText)              'find location of passed delimiter

    Do While delLoc > 0                                 'only bother if there is a delimiter
        docText = Left(docText, delLoc - delLen) & replaceIt & _
                  Right(docText, Len(docText) - delLoc - delLen + 1) 'rebuild string with
replaced or deleted delimiter
        delLoc = InStr(docText, delimiterText)          'find location of next delimiter in title
    Loop                                                'finished with this section
    stripDelimiters = Trim(docText)                     'strip leading and trailing spaces
End Function

Public Function convertCharacter(ByVal inputString As String, _
                                 ByVal v As String, _
                                 ByVal r As String) As String
'***
'***    This function takes the passed input string and replaces all occurances of
'***    character "v" passed to it with the character in "r".
'***
    Dim vLoc As Long                                           'delimiter location in string
    Dim outputString As String                                 'place to build output string
    Dim s As Long                                              'continue search from here
    '
    outputString = ""
    s = 1
    '
    vLoc = InStr(inputString, v)                       'find location of passed delimiter

    Do While vLoc > 0                                  'only bother if there is a delimiter
        outputString = outputString & Mid(inputString, s, vLoc - s) & r 'build output string
with "v" replaced with "r"
        s = vLoc + 1
        vLoc = InStr(s, inputString, v)   'find location of next "v" in string, start looking
after last "v"
    Loop                                  'finished with this section
    outputString = outputString & Right(inputString, Len(inputString) - s + 1)
    convertCharacter = outputString       'strip leading and trailing spaces
End Function
```

```
'=======================================================================
Public Sub InsertBottomText()
Dim bottomString As String
Dim index1String As String
Dim index2String As String
Dim mainIndexString As String
Dim HIILegalString As String

bottomString = getFile(conversionsFolder & ":" & bottomText)
index1String = getFile(conversionsFolder & ":" & index1Return)
mainIndexString = getFile(conversionsFolder & ":" & mainIndexReturn)
HIILegalString = getFile(conversionsFolder & ":" & HIILegal)

    Selection.HomeKey unit:=wdStory
    SimpleFind docBottomToken                'find where to stick bottom text
    Selection.Collapse wdCollapseEnd
    Selection.Text = vbCr & bottomString

    Selection.HomeKey unit:=wdStory
    SimpleFind index1Token
    Selection.Collapse wdCollapseEnd
    Selection.Text = index1String

    Selection.HomeKey unit:=wdStory
    SimpleFind mainIndexToken
    Selection.Collapse wdCollapseEnd
     Selection.Text = mainIndexString

    Selection.HomeKey unit:=wdStory
    SimpleFind HIILegalToken
    Selection.Collapse wdCollapseEnd
    Selection.Text = HIILegalString

    Selection.HomeKey unit:=wdStory
    ReplaceEvery moduleToken, moduleGif
    Selection.Collapse wdCollapseEnd

    If secondIndex Then
        Selection.HomeKey unit:=wdStory
        index2String = getFile(conversionsFolder & ":" & index2Return)
        SimpleFind index2Token
        Selection.Collapse wdCollapseEnd
        Selection.Text = index2String
        Selection.HomeKey unit:=wdStory
        ReplaceEvery moduleToken2, moduleGif2
        Selection.Collapse wdCollapseEnd
    End If
End Sub

'=============================================================
Public Sub InsertTopText()
Dim topString As String

topString = getFile(conversionsFolder & ":" & topText)

    Selection.HomeKey unit:=wdStory
    SimpleFind docTopToken                  'find where to stick top text
    Selection.Collapse wdCollapseEnd
    Selection.Text = vbCr & topString

    Selection.HomeKey unit:=wdStory
    SimpleFind cTag
    If Selection.Find.Found Then
        Selection.Text = ASPdochdr1 & vbCr & cTag  'insert includes file ref
        Selection.Collapse wdCollapseEnd
    End If

    Selection.HomeKey unit:=wdStory
    ReplaceEvery moduleBackToken, moduleBack
    Selection.Collapse wdCollapseEnd
End Sub

Public Function getFile(ByVal fileName As String) As String
'***
'***    Read a complete file into a string
'***
Dim fileNumber As Long
Dim inputString As String

    getFile = ""
    fileNumber = FreeFile                           'get next available file number
    Open fileName For Input As #fileNumber          'open external file for read
    Do While Not EOF(fileNumber)
        Line Input #fileNumber, inputString              'read in a line
        getFile = getFile & inputString & vbCr      'build result string
    Loop
    Close #fileNumber                               'close file
End Function

Public Sub tweakModuleName()
'***
'***    Take entered module name and fix name,
'***    determine if there is a second index
'***    and set lowercase name.
'***

    moduleGif2 = "none"        'default to no gif 2

    Select Case moduleName
        Case "AHA"
            secondIndex = False
            moduleName = "AHA"
            moduleGif = "aha"
            moduleBack = "ahaback"
        Case "PA"
            secondIndex = False
            moduleName = "PedAdv"
            moduleGif = "pa"
            moduleBack = "paback"
       Case "BHA"
            secondIndex = True
            moduleName = "BHA"
            moduleGif = "bha1"
            moduleGif2 = "bha2"          'only BHA has a second index in second gif (for now)
            moduleBack = "bhaback"
        Case "WHA"
            secondIndex = False
            moduleName = "WHA"
            moduleGif = "wha"
            moduleBack = "whaback"
        Case "SHA"
            secondIndex = False
            moduleName = "SHA"
            moduleGif = "sha"
            moduleBack = "shaback"
       Case "CA"
            secondIndex = False
            moduleName = "CA"
            moduleGif = "ca"
            moduleBack = "caback"
        Case "OA"
            secondIndex = False
            moduleName = "OA"
            moduleGif = "oa"
            moduleBack = "oaback"
        Case "SMA"
            secondIndex = False
            moduleName = "SMA"
            moduleGif = "sma"
            moduleBack = "smaback"
        Case "MA"
            secondIndex = False
            moduleName = "MedAdv"
            moduleGif = "ma"
            moduleBack = "maback"
        Case Else
            secondIndex = False
            moduleName = "DebugMePlease"
            moduleGif = "debugMe"
            moduleBack = "debugmeback"
    End Select
End Sub

'==========================================================================
Option Explicit
Public Sub Make2ColumnList()

    preformattedText beg2ColList, end2ColList    'add preformatting start end end tags to 2
column lists
End Sub

'==========================================================================
Option Explicit
Public Sub MakeBookList()

    preformattedText begBookList, endBookList    'add preformatting start end end tags to
booklists
End Sub

'==========================================================================
Option Explicit
Public Sub MakeLists()
'***
'***    This routine looks for the various list tags and processes accordingly.
'***    Real html lists are created whenever possible.
'***

    preItemList numTag          'use numbered item list preformatting
    preItemList letTag          'use letter item list preformatting
    preItemList romTag          'use roman numeral item list preformatting
    preItemList bliTag          'use bulleted item list preformatting, except for simple case
                                'if simple bulleted list, create actual html list
    bliItems "3/0"              'prettify simple bulleted lists
'   bliItems "7/4" 'bliItems needs to be debugged for this case, leave as "pre" text for now
End Sub

Public Sub preItemList(ByVal listType As String)
'***
'***    This routine searched for the types of list passed to it and simply leaves it
'***    reformatted.  This is ugly, but works for the time being.
'***
Dim paraInfo As String        'contains the n/m indent information  which goes with list tag
Dim bParaTag As String      'contains html string which is inserted at the beginnning of list
Dim eParaTag As String      'contains html string which is inserted at the end of list
Dim listItem As String      'holds a copy of selected list text to process

    Selection.HomeKey unit:=wdStory                      'always know where you are (top)
    SimpleFind vbCr & listType                    'find start of list tag
    Do While Selection.Find.Found
        Selection.MoveLeft  'leaves insertion point before hard return of previous paragraph
        Selection.MoveDown unit:=wdParagraph              'move to beginning of tag line
        Selection.InsertAfter htmlStartComment            'want to comment out tag
        Selection.MoveRight Count:=Len(listType)          'move to right of tag
        Selection.MoveDown unit:=wdParagraph, Extend:=wdExtend     'select x/x info
        Selection.End = Selection.End - 1                 'deselect hard return

        paraInfo = Trim(Selection.Text)                   'save cleaned selection

        Selection.Collapse wdCollapseEnd                  'deselect tag
        Selection.InsertAfter htmlEndComment              'finish commenting out tag
        Selection.MoveDown unit:=wdParagraph          'move to beginning of body of list item
        If listType = bliTag Then                  'if simple bulleted list, do more processing
            Select Case paraInfo
                Case "3/0"
                    bParaTag = "<li>"
                    eParaTag = " "
                    Selection.MoveRight Extend:=wdExtend, Count:=3   'select dash and spaces
                    Selection.Range.Delete                         'delete same
'                Case "7/4"
'                    bParaTag = "<li>"
'                    eParaTag = " "
'                    Selection.MoveRight Extend:=wdExtend, Count:=7  'select dash and spaces
'                    Selection.Range.Delete                         'delete same
                Case Else
                    bParaTag = htmlParaStart & htmlPreStart & vbCr 'indented paragraph of some
other sort, use as is
                    eParaTag = vbCr & htmlPreEnd & htmlParaEnd  'leave as is until I have more
```

```
time
            End Select
        Else
            bParaTag = htmlParaStart & htmlPreStart & vbCr 'indented paragraph of some other
sort, use as is
            eParaTag = vbCr & htmlPreEnd & htmlParaEnd    'leave as is until I have more time
        End If

        Selection.InsertAfter bParaTag                    'insert html para tag
        Selection.CollapsewdCollapseEnd
        SimpleFind paraDelimiter                     'find end of list item paragraph
        If Selection.Find.Found Then
            Selection.Text = eParaTag                'replace token with html closing para tag
            Selection.Collapse wdCollapseEnd
        Else
            Selection.MoveUntil cset:="/" 'move insertion point until the next beginning of a
/tag is found
            Selection.MoveLeft                           'move to end of previos
            Selection.InsertAfter missingParaDelimiterDebug & eParaTag 'insert a debug msg
because end of list item tag wasn't found
            Selection.Collapse wdCollapseEnd
        End If
        SimpleFind vbCr & listType                    'find start of next list tag
    Loop
End Sub

Public Sub bliItems(ByVal indent As String)
'***
'***    now that list item are marked, figure out which list items go together
'***
Dim paraInfo As String         'contains the n/m indent information  which goes with list tag
Dim bParaTag As String      'contains html string which is inserted at the beginnning of list
Dim eParaTag As String      'contains html string which is inserted at the end of list
Dim listItem As String      'holds a copy of selected list text to process
Dim hpLoc As Long           'holds location of /hp tag in listItem

    Selection.HomeKey unit:=wdStory                       'always know where you are (top)
    SimpleFind htmlStartComment & bliTag & indent          'start of list
    Do While Selection.Find.Found
        Selection.MoveLeft
        Select Case indent
            Case "3/0"
        Selection.InsertAfter "<ul>" & vbCr                'html tag for start of list
'             Case "7/4"
'         Selection.InsertAfter htmlBQStart & vbCr & "<ul>" & vbCr 'html tag for start of list
            Case Else
        End Select
        Selection.MoveDown unit:=wdParagraph               'prevent infinite loop if no /p
        Selection.ExtendMode = True                        'get tricky
        SimpleFind pCommentTag                            'find end of list - assume next para
        If Selection.Find.Found Then
            listItem = Selection.Text                      'save selection to manipulate
            hpLoc = InStr(listItem, hpCommentTag) 'find first occurance of primary header, if
any
            Selection.ExtendMode = False
            If hpLoc = 0 Then  'no precluding /hp - an /hp tag would occur after end of list
                Selection.Collapse wdCollapseEnd 'collapse selection to end after pCommentTag
                Selection.MoveLeft Count:=Len(pCommentTag) 'move insertion point to before
pCommentTag to insert </ul>
            Else
                Selection.Collapse wdCollapseStart        'go back before "<!-- /bli" tag
                SimpleFind hpCommentTag         'find end of list - we know it is <!-- /hp -->
                Selection.MoveLeft                        'now we're in the right spot
            End If
        Else                             'no pCommentTag found to end list, look for next header
            Selection.ExtendMode = False

            SimpleFind hpCommentTag                'find end of list - assume next header
            If Selection.Find.Found Then
                Selection.Collapse wdCollapseStart        'right spot to insert </ul>
            Else
                SimpleFind htmlStartComment & bliTag & indent        'find next list item

                Do While Selection.Find.Found
                    SimpleFind htmlStartComment & bliTag & indent 'keep finding list items,
there are no intervening /p or /hp
                Loop
                Selection.MoveDown unit:=wdParagraph      'start next find in next paragraph
                SimpleFind htmlStartComment       'find next tag as assumed end to list
                Selection.Collapse wdCollapseStart        'collapse tostart of tag
            End If
        End If

        '***
        '***    At this point we are at the presumed end of the simple bulleted list.
        '***

        Select Case indent
            Case "3/0"
        Selection.InsertAfter "</ul>" & vbCr                'html tag for end of list
            Case "7/4"
        Selection.InsertAfter "</ul>" & vbCr & htmlBQEnd & vbCr    'html tag for end of list
            Case Else
        End Select

        Selection.MoveDown unit:=wdParagraph                'prevent infinite loop if no /p, /hp
        SimpleFind htmlStartComment & bliTag & indent         'find start of next list
    Loop
End Sub

'===========================================================================
Option Explicit
Public Sub MakeMenu()

    preformattedText begMenu, endMenu     'add preformatting start end end tags to menus
End Sub

'===========================================================================
Option Explicit
Public Sub MakeParagraphs()
'***
'***    Change /p tags into html paragraphs
'***
    Dim paraInfo As String                                'info after the /p, e.g. 0/0
Dim bParaTag As String
Dim eParaTag As String

Selection.HomeKey unit:=wdStory                             'always know where you are
SimpleFind vbCr & pTag                              'start of para tag
If Not Selection.Find.Found Then
    Selection.InsertAfter pTagDebug                         'what? no /p? add debug message
    Selection.Collapse wdCollapseEnd
    Exit Sub                                                'don't bother with rest
End If
Do While Selection.Find.Found
    Selection.MoveLeft
    Selection.MoveDown unit:=wdParagraph
    Selection.InsertAfter htmlStartComment                  'want to comment out tag
    Selection.MoveRight Count:=3                            'move to right of tag
    Selection.MoveDown unit:=wdParagraph, Extend:=wdExtend      'select x/x info
    Selection.End = Selection.End - 1                       'deselect hard return

    paraInfo = Selection.Text                               'save selection
    paraInfo = Trim(paraInfo)                               'clean up info

    Selection.MoveRight
    Selection.InsertAfter htmlEndComment                    'finish comment
    Selection.MoveDown unit:=wdParagraph

    Select Case paraInfo
        Case "0/0"                                          'simple paragraph
            bParaTag = htmlParaStart
            eParaTag = htmlParaEnd
        Case "2/0"                                          'simple paragraph
            bParaTag = htmlParaStart
            eParaTag = htmlParaEnd
        Case "0/4"                                          'simple paragraph
            bParaTag = htmlParaStart
            eParaTag = htmlParaEnd
        Case "4/4"                                          'indented paragraph
            bParaTag = htmlBQStart                          'use blockquote to simulate indent
            eParaTag = htmlBQEnd
        Case "4/5"                                          'indented paragraph
            bParaTag = htmlBQStart                          'use blockquote to simulate indent
            eParaTag = htmlBQEnd
        Case Else
            bParaTag = htmlParaStart & htmlPreStart         'indented paragraph of some sort
            eParaTag = htmlPreEnd & htmlParaEnd             'leave as is until I have more time
    End Select

    Selection.InsertAfter bParaTag                          'insert html para tag
    Selection.Collapse wdCollapseEnd
    SimpleFind paraDelimiter                        'find token for end of paragraph, namely ""
    If Selection.Find.Found Then
        Selection.Text = eParaTag                       'replace token with html closing para tag
        Selection.Collapse wdCollapseEnd
    Else                                                    'geeze, where's the delimiter?
        Selection.InsertAfter missingParaDelimiterDebug 'insert yet a nother debug comment
        Selection.Collapse wdCollapseEnd
    End If
    SimpleFind vbCr & pTag                          'find next paragraph
    Loop
End Sub

'=========================================================================
Option Explicit
Public Sub MakeTables()

    preformattedText begTable, endTable     'add preformatting start end end tags to tables
End Sub

'=========================================================================
Option Explicit
Public Sub multiReplace(ByVal pairsFile As String)
'
'***
'***    Read pairs of find/replace strings from an external file and perform ReplaceAll
'***    Intended to replace the Add/Strip steps of the original conversion procedure.
'***
'***    pairsFile   =>  filename of the file containing the find/replace pairs
'***
'***    file format is as follows:
'***
'***        x,y,z,search,replace
'***
'***    where:
'***            x   =>  "1" or "0"      mapped to "enabled", find/replace is skipped if
'***            enabled = "0"
'***            y   =>  "1" or "0"      mapped to "caseSensitive", case is ignored in
'***            search/replace if caseSensitive = "0"
'***            z   =>  "1" or "0"      mapped to "useWildCards", wildcard search/replace is
'***            enabled if useWildCards = "1"
'***            search   =>  search string    mapped to "findString"
'***            replace  =>  replace string   mapped to "replaceString"
'***
'***    note:   if "findString" or "replaceString" have embedded double quotes <"> or commas
'***            <,>, they appear in the conversion file as a bar character <|> or bullet <>,
'***            respectively.  The bar and bullet characters were used in the conversion file
'***            since the since the double-quote and comma characters are delimiters for the
'***            file read function.  The "convertCharacter" function fixes this.
'***
'

Dim fileNumber As Long
Dim enabled As String
Dim caseSensitive As String
Dim useWildCards As String
Dim findString As String
Dim replaceString As String

    Reset                                       'make sure all low level access files are closed
    fileNumber = FreeFile                                   'get next available file number
'    Debug.Print pairsFile
'    Debug.Print fileNumber
'    Debug.Print CurDir
    Open pairsFile For Input As #fileNumber               'open external file for read
    Do While Not EOF(fileNumber)
        Input #fileNumber, enabled, caseSensitive, _
                useWildCards, findString, replaceString   'read in a find/replace pair
        findString = convertCharacter(findString, "|", Chr(34))        'make quotes real
        replaceString = convertCharacter(replaceString, "|", Chr(34)) 'make quotes real
        findString = convertCharacter(findString, "", ",")        'make commas real
        replaceString = convertCharacter(replaceString, "", ",") 'make commas real

'Debug.Print enabled
'Debug.Print caseSensitive
'Debug.Print useWildCards
'Debug.Print findString
'Debug.Print replaceString
'Debug.Print "-------------------------------------------------"
        If enabled = "1" Then
   'call ReplaceAll method
            ActiveDocument.Content.Find.ClearFormatting
            ActiveDocument.Content.Find.Replacement.ClearFormatting

            With ActiveDocument.Content.Find
                .Text = findString
                .Replacement.Text = replaceString
                .Forward = True
                .Wrap = wdFindContinue
                .Format = False
                If caseSensitive = "1" Then
                    .MatchCase = True
                Else
                    .MatchCase = False
                End If
                .MatchWholeWord = False
                If useWildCards = "1" Then
                    .MatchWildcards = True
                Else
                    .MatchWildcards = False
                End If
                .MatchSoundsLike = False
                .MatchAllWordForms = False
                .Execute Replace:=wdReplaceAll
            End With
        End If
    Loop
        Close #1    'close file
End Sub

'====================================================================
Option Explicit
Public Sub PreformattedListsandTables()

    preformattedText beg2ColList, end2ColList 'add preformatting start end end tags to 2
column lists
    preformattedText begBookList, endBookList 'add preformatting start end end tags to
booklists
    preformattedText begMenu, endMenu    'add preformatting start end end tags to menus
    preformattedText begTable, endTable  'add preformatting start end end tags to tables
End Sub


'====================================================================
Option Explicit
```

```
Public Sub SectionLinks()
'***
'***    Creates internal document links based upon
'***    section titles started by  "/hp" tag aka "hpTag".
'***
    Dim linkCount As Long           'keeps a running count as to how many links have been made
    Dim sectionTitle As String        'title of a subsection used to make a link at top of doc
    Dim linkIndex As String                               'holds anchor index name
    Dim paraLoc As Long

    linkCount = 0
    sectionTitle = ""
    linkIndex = ""

    Selection.HomeKey unit:=wdStory                       'start at the top
    SimpleFind hpCommentTag                               'find first primary heading
    Do While Selection.Find.Found
        Selection.MoveDown unit:=wdParagraph              'move to beginning of next paragraph
        Selection.ExtendMode = True
        SimpleFind "^p/"                                  'select to next  if found
        If Selection.Find.Found Then                      ' "^p/" was found
            Selection.End = Selection.End - 2             'shrink selection to just title text
            Selection.ExtendMode = False
            linkCount = linkCount + 1                     'point to next link, starts at "1"
            linkIndex = Format(linkCount)  'make count into a string, use Format() for no
leading space
            sectionTitle = Selection.Text          'copy section title w/"" characters if any
            sectionTitle = stripDelimiters(sectionTitle, paraDelimiter, " ") 'clean section
title of "" characters
            sectionTitle = stripDelimiters(sectionTitle, vbCrLf, " ") 'clean section title of
hard return characters
            sectionTitle = stripDelimiters(sectionTitle, vbCr, " ") 'clean section title of
hard return characters

    '***
    '*** insert the new text including tokens where html will be added in place of selection
    '***

            Selection.Text = htmlBreak & startAnchorToken & linkIndex & _
                             endAnchorToken & sectionTitle & htmlParaEnd & hpLastTag
            Selection.Collapse wdCollapseEnd
            Selection.HomeKey unit:=wdStory 'go to top of doc to insert link to just added
anchor
            SimpleFind TopLinkTag                         'this is where the top links go
            Selection.Text = htmlTopLinkStart & linkIndex & qq & _
                            htmlET & sectionTitle & htmlAEnd & _
                            vbCr & TopLinkTag 'insert actual link, leave top link for next one
            Selection.Collapse wdCollapseEnd
            SimpleFind hpLastTag                          'go back to where we left off
            Selection.Text = ""                           'delete last seen tag
        End If
        SimpleFind hpCommentTag                           'find next primary heading
    Loop                                                  'while primary header found
    Selection.HomeKey unit:=wdStory                       'go back to top
    SimpleFind TopLinkTag                     'go back to end of last inserted link at top

    If Selection.Find.Found Then                 'make sure there were some links added
```

```
        Selection.Text = vbCr & vbCr
        Selection.Collapse wdCollapseEnd
        Selection.Text = ASPdochdr2 & vbCr                    'include dochdr2 for asp
        Selection.Collapse wdCollapseEnd
    Else
        SimpleFind afterTitleTag                              'must assume this exists
        Selection.MoveDown unit:=wdParagraph                  'move down to after main title
        Selection.Collapse wdCollapseEnd
        Selection.Text = vbCr & ASPdochdr2 & vbCr             'include dochdr2 for asp
        Selection.Collapse wdCollapseEnd
    End If
    Selection.HomeKey unit:=wdStory                           'leave at top of doc
End Sub

'============================================================================
'============================================================================
'===    The followiung is the macro which is used to convert indexes
'===    Some of the subroutines and functions have the same names as
'===    earlier referenced routines, but they are private to this macro set
'===    Listed as independent macro for temporary development use to facilitate
'===    Debugging and to permit slightly altered code in some subroutines
'============================================================================
Option Explicit
Public indexName As String
Public index2Name As String
Public workingDirectory As String
Public targetFolder As String
Public conversionsFolder As String
Public moduleName As String
Public moduleGif As String
Public moduleGif2 As String
Public moduleBack As String    'name of background gif
Public secondIndex As Boolean

Public currentFile As String   'so we only have to check once
Public currFileWindow As String
'Public linkMapWindow As String
'Public docMapWindow As String

Public namePart As String
Public currAlpha As String

Public Sub prepIndex()
'***
'***    set up parameters
'***
Application.DisplayScrollBars = False
'Application.DisplayStatusBar = False
Application.ScreenUpdating = False
    Dim currFile As String
    Dim extPart As String
    Dim saveName As String
    Dim messageText As String
    Dim moduleCount As Long

    currFile = ""
    saveName = ""
    namePart = ""
    extPart = ""

    messageText = "Enter module code:" & vbCr & _
                        "AHA - Adult Health Advisor" & vbCr & _
                        "PA - Pediatric Health Advisor" & vbCr & _
                        "BHA - Behavioural Health Advisor" & vbCr & _
                        "WHA - Women's Health Advisor" & vbCr & _
                        "SHA - Senior Health Advisor" & vbCr & _
                        "CA - Cardiac Health Advisor" & vbCr & _
                        "OA - Ophthalmology Health Advisor" & vbCr & _
                        "SMA - Sports Medicine Advisor" & vbCr & _
                        "MA - Medications Advisor" & vbCr

    moduleName = InputBox(messageText, "Module Name Entry", "AHA")
    tweakModuleName                          'set some globals based on entered module name
    workingDirectory = getArticlePath()      'get working directory which contains raw articles
    targetFolder = makeTargetFolder(moduleName) 'create a folder to save converted articles
and indexes
    conversionsFolder = setConversionsFolder("Conversions")

    currFile = Dir("") '(workingDirectory)                'retrieve first file
    currFile = StrConv(currFile, vbLowerCase)

    Do While currFile <> ""                 'null string is returned when no more files in folder
        namePart = getNamePart(currFile)    'seperate the current filename into namepart and
extension part
        extPart = getExtPart(currFile)
        ChDir workingDirectory
        Documents.Open _
            fileName:=currFile, _
            ConfirmConversions:=False, _
            addToRecentFiles:=False, _
            Format:=wdOpenFormatText          'open current file for processing
'        currentFile = ActiveDocument.Name    'get current filename sans path
        convertIndex                                      'convert as index file
        Documents(currFile).Close savechanges:=wdDoNotSaveChanges
        currFile = Dir                                    'get next file
        currFile = StrConv(currFile, vbLowerCase)
    Loop

Application.DisplayScrollBars = True
'Application.DisplayStatusBar = True
Application.ScreenUpdating = True
End Sub

Private Sub convertIndex()
Dim linkText As String
Dim saveName As String
Dim n As Long
Dim newSubIndex As Document

    removeNonIndexLines                      'get rid of extraneous stuff
    For n = 65 To 90
        currAlpha = Chr(n)
        linkText = getCurrAlphaLinks
        Set newSubIndex = Documents.Add
        Documents(newSubIndex).Activate
        Selection.Text = linkText
        Selection.Collapse wdCollapseStart
        formatIndexLinks
        Selection.HomeKey unit:=wdStory
        InsertTopText
        InsertBottomText
        ReplaceEvery "^p^p^p", vbCr
        saveName = targetFolder & ":" & namePart & "_" & StrConv(currAlpha, vbLowerCase) &
".asp"
        With Documents(newSubIndex)
            .SaveAs fileName:=saveName, FileFormat:=wdFormatText
            .Close
        End With
    Next n
End Sub

Private Sub formatIndexLinks()
Dim link As String
Dim tildeLoc As Long
Dim linkLen As Long

'***
'***    Clean up non link lines
'***
Selection.HomeKey unit:=wdStory
ReplaceEvery "~^p", "^p"

'***
'***    format link lines
'***
Selection.HomeKey unit:=wdStory
SimpleFind "~"
Do While Selection.Find.Found
    WordBasic.Insert "<!-- ~ -->"
    Selection.MoveDown unit:=wdParagraph, Extend:=wdExtend
    Selection.End = Selection.End - 1
    link = Selection.Text
    Selection.Text = ""
    tildeLoc = InStr(link, "~")
    linkLen = Len(link)
    link = Left(link, tildeLoc - 1) & ".asp#" & Right(link, linkLen - tildeLoc)
    link = "<a href=" & Chr(34) & link & Chr(34) & ">"
    Selection.MoveUp unit:=wdParagraph
    Selection.Text = link
    Selection.Collapse wdCollapseEnd
    Selection.ExtendMode = True
    SimpleFind "<!-- ~ -->"
    Selection.ExtendMode = False
    Selection.End = Selection.End - 10
    Selection.Text = RTrim(Selection.Text)
    Selection.Collapse wdCollapseEnd
    Selection.Text = "</a>"

    Selection.Collapse wdCollapseEnd
    Selection.End = Selection.End + 10
    Selection.Text = ""
    SimpleFind "~"
Loop
End Sub

'***
'***    just search forward for the passed string
'***
Private Sub SimpleFind(ByVal findStr As String)

    With Selection.Find
        .Text = findStr
        .Forward = True
        .Format = False
        .MatchAllWordForms = False
        .MatchCase = False
        .MatchSoundsLike = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .Wrap = wdFindStop
        .Execute
    End With
End Sub

'***
'***    search and replace forward using the passed strings for all occurances
'***

Private Sub ReplaceEvery(ByVal findStr As String, ByVal replaceStr As String)
    Selection.HomeKey unit:=wdStory
    With Selection.Find
        .Text = findStr
        .Replacement.Text = replaceStr
        .Forward = True
        .Format = False
        .MatchAllWordForms = False
        .MatchCase = False
        .MatchSoundsLike = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .Wrap = wdFindContinue
        .Execute Replace:=wdReplaceAll
    End With
    Selection.HomeKey unit:=wdStory
End Sub

Private Sub tweakModuleName()
'***
'***    Take entered module name and fix name,
'***    determine if there is a second index
'***    and set lowercase name.
'***

    moduleGif2 = "none"         'default to no gif 2
    index2Name = "none"         'default to no index 2

    Select Case moduleName
        Case "AHA"
        secondIndex = False
        moduleName = "AHA"
        moduleGif = "aha"
        moduleBack = "ahaback"
        indexName = "Adult Health Advisor Index"
    Case "PA"
        secondIndex = False
        moduleName = "PedAdv"
        moduleGif = "pa"
        moduleBack = "paback"
        indexName = "Pediatric Advisor Index"
    Case "BHA"
        secondIndex = True
        moduleName = "BHA"
        moduleGif = "bha1"
        moduleGif2 = "bha2"         'only BHA has a second index in second gif (for now)
        moduleBack = "bhaback"
        indexName = "Behavioral Health Advisor Adult Index"
        index2Name = "Behavioral Health Advisor Pediatric Index"
    Case "WHA"
        secondIndex = False
        moduleName = "WHA"
        moduleGif = "wha"
        moduleBack = "whaback"
        indexName = "Women's Health Advisor Index"
    Case "SHA"
        secondIndex = False
        moduleName = "SHA"
        moduleGif = "sha"
        moduleBack = "shaback"
        indexName = "Senior Health Advisor Index"
    Case "CA"
        secondIndex = False
        moduleName = "CA"
        moduleGif = "ca"
        moduleBack = "caback"
        indexName = "Cardiac Advisor Index"
    Case "OA"
        secondIndex = False
        moduleName = "OA"
        moduleGif = "oa"
        moduleBack = "oaback"
        indexName = "Ophthalmology Advisor Index"
    Case "SMA"
        secondIndex = False
        moduleName = "SMA"
        moduleGif = "sma"
        moduleBack = "smaback"
        indexName = "Sports Medicine Advisor Index"
    Case "MA"
        secondIndex = False
        moduleName = "MedAdv"
        moduleGif = "ma"
        moduleBack = "maback"
        indexName = "Medications Index"
    Case Else
        secondIndex = False
```

```
            moduleName = "DebugMePlease"
            moduleGif = "debugMe"
            moduleBack = "debugmeback"
            indexName = "Debug Me Please Index"
    End Select
End Sub

Private Function getArticlePath() As String
'***
'***    The purpose of this function is to return the path
'***    for the articles to be converted
'***
'
    With Dialogs(wdDialogFileOpen)
        .Show
    End With

    '***
    '***    it doesn't matter if a file is opened or not,
    '***    CurDir returns the last navigated path
    '***    return value to calling
    '***

    getArticlePath = CurDir
End Function

Private Function getNamePart(ByVal fileName As String) As String
'***
'***    Return the name part of a filename
'***
    Dim dotLoc As Long
    dotLoc = InStr(fileName, ".")
    getNamePart = Left(fileName, dotLoc - 1)
End Function

Private Function getExtPart(ByVal fileName As String) As String
'***
'***    Return the extension part of a filename
'***
    Dim dotLoc As Long
    dotLoc = InStr(fileName, ".")
    getExtPart = Right(fileName, Len(fileName) - dotLoc)
End Function

Private Function makeTargetFolder(ByVal targetFolder As String) As String
'***
'***    MkDir ==> create a folder based on the current path
'***
'***    The "::" part makes it go up one level before creating the directory
'***
Dim saveDir As String
    MkDir "::" & targetFolder      'create target folder on the same level as working folder
    saveDir = CurDir      'save current folder to return to
    ChDir "::" & targetFolder      'go to newly created folder
    makeTargetFolder = CurDir 'make the return value of this function be the created folder
    ChDir saveDir         'go back to working folder
End Function
```

```
Private Function setConversionsFolder(ByVal cFolder As String) As String
'***
'***    MkDir ==> create a folder based on the current path
'***
'***    The "::" part makes it go up one level before creating the directory
'***
Dim saveDir As String
    saveDir = CurDir        'save current folder to return to
    ChDir "::" & cFolder        'go to newly created folder
    setConversionsFolder = CurDir 'make the return value of this function be the created
folder
    ChDir saveDir           'go back to working folder
End Function

Private Sub saveSubIndex()
                    saveName = targetFolder & ":" & namePart & "_" & currAlpha & ".asp"
                    With Documents(newSubIndex)
                        .SaveAs fileName:=saveName, FileFormat:=wdFormatText
                        .Close
                    End With
End Sub

Private Sub removeNonIndexLines()
'***
'***    Clean up PC formatted lines
'***
Selection.HomeKey unit:=wdStory
ReplaceEvery vbCrLf, "^p"
'***
'***    remove extraneous text, leave only index lines
'***
Selection.HomeKey unit:=wdStory
Selection.ExtendMode = True
SimpleFind "]"
Selection.MoveDown unit:=wdParagraph
Selection.ExtendMode = False
Selection.Text = ""
'***
'***    deleted everything at the top at this point
'***
SimpleFind "^p.Q"
Selection.Text = ""

'***
'***    deleted everything at the bottom at this point
'***
End Sub

Private Sub InsertBottomText()
Dim bottomString As String
Dim index1String As String
Dim index2String As String
Dim mainIndexString As String
Dim HIILegalString As String
bottomString = getFile(conversionsFolder & ":" & "subIndexBottom.txt")
index1String = getFile(conversionsFolder & ":" & "index1Return.txt")
mainIndexString = getFile(conversionsFolder & ":" & "mainIndexReturn.txt")
```

```
HIILegalString = getFile(conversionsFolder & ":" & "HIILegal.txt")
    Selection.EndKey unit:=wdStory              'go to bottom
    Selection.Text = vbCr & bottomString
    Selection.HomeKey unit:=wdStory
    SimpleFind "<!-- index1Return -->"
    Selection.Collapse wdCollapseEnd
    Selection.Text = index1String
    Selection.HomeKey unit:=wdStory
    SimpleFind "<!-- mainIndexReturn -->"
    Selection.Collapse wdCollapseEnd
    Selection.Text = mainIndexString
    Selection.HomeKey unit:=wdStory
    SimpleFind "<!-- HIILegal -->"
    Selection.Collapse wdCollapseEnd
    Selection.Text = HIILegalString
    Selection.HomeKey unit:=wdStory
    ReplaceEvery "$module$", moduleGif
    Selection.Collapse wdCollapseEnd
    If secondIndex Then
        Selection.HomeKey unit:=wdStory
        index2String = getFile(conversionsFolder & ":" & "index2Return.txt")
        SimpleFind "<!-- index2Return -->"
        Selection.Collapse wdCollapseEnd
        Selection.Text = index2String
        Selection.HomeKey unit:=wdStory
        ReplaceEvery "$module2$", moduleGif2
        Selection.Collapse wdCollapseEnd
    End If
End Sub

Public Sub InsertTopText()
Dim topString As String
topString = getFile(conversionsFolder & ":" & "subIndexTop.txt")
    Selection.HomeKey unit:=wdStory
    Selection.Text = topString & vbCr
    Selection.HomeKey unit:=wdStory
    If namePart = "index" Then
        ReplaceEvery "$indexName$", indexName
    Else
        ReplaceEvery "$indexName$", index2Name
    End If
    Selection.Collapse wdCollapseEnd
    Selection.HomeKey unit:=wdStory
    ReplaceEvery "$moduleback$", moduleBack
    Selection.Collapse wdCollapseEnd
    Selection.HomeKey unit:=wdStory
    ReplaceEvery "$A$", currAlpha
    Selection.Collapse wdCollapseEnd
End Sub

Public Function getFile(ByVal fileName As String) As String
'***
'***    Read a complete file into a string
'***
Dim fileNumber As Long
Dim inputString As String
    getFile = ""
    fileNumber = FreeFile                                       'get next available file number
    Open fileName For Input As #fileNumber                      'open external file for read
    Do While Not EOF(fileNumber)
        Line Input #fileNumber, inputString                          'read in a line
        getFile = getFile & inputString & vbCr                  'build result string
    Loop
    Close #fileNumber                                           'close file
End Function

Private Function getCurrAlphaLinks() As String
    Dim workingString As String
    Dim foundNext As Boolean
    Dim nextAlpha As String
    workingString = ""
    foundNext = False
    nextAlpha = Chr(Asc(currAlpha) + 1)
    Selection.HomeKey unit:=wdStory
    SimpleFind "^p" & currAlpha
    If Not Selection.Find.Found Then                            'if you don't find it then git!
        getCurrAlphaLinks = vbCr & vbCr & vbCr & "Sorry, there are no topics " & _
                            "which begin with the letter " & _
                            Chr(34) & currAlpha & Chr(34) & "."
        Exit Function
    End If
    Do While Selection.Find.Found
        Selection.ExtendMode = True
        SimpleFind "^p" & currAlpha
    Loop
    workingString = workingString & Selection.Text
    Selection.Text = ""
    Selection.ExtendMode = False
    '***
    '***    need to make sure to handle indented section after last currAlpha
    '***
    Selection.ExtendMode = True
    Do Until foundNext Or Asc(nextAlpha) > 90
        SimpleFind "^p" & nextAlpha
        If Selection.Find.Found Then
            foundNext = True
            Selection.ExtendMode = False
            Selection.End = Selection.End - 2                   'deselect character and hard return
            workingString = workingString & Selection.Text
            Selection.Text = ""
        Else
            nextAlpha = Chr(Asc(nextAlpha) + 1)
        End If
    Loop
    If Asc(nextAlpha) > 90 Then
        Selection.EndKey unit:=wdStory
        workingString = workingString & Selection.Text
        Selection.Text = ""
    End If
    Selection.ExtendMode = False
    getCurrAlphaLinks = workingString
End Function
```